FONDAMENTI DI PROGRAMMAZIONE
CORSO DI LAUREA IN INGEGNERIA INFORMATICA

PROVA PRATICA 26 GIUGNO 2023

Un `Razionale` rappresenta un numero razionale. Esso è composto da un numeratore e un denominatore, entrambi interi. Ovviamente, un numero razionale può essere negativo o zero.

$4 = \frac{4}{1}$ $\quad 0.50 = \frac{1}{2}$ $\quad 0.3\overline{3} = \frac{1}{3}$

Implementare le seguenti operazioni che possono essere effettuate su un `Razionale`. **È espressamente vietato l'utilizzo di variabili in virgola mobile.**

--- Metodi invocati nella PRIMA PARTE di main.cpp: ---

✓ **`Razionale r;`**

Costruttore di default che inizializza un `Razionale` uguale a zero.

✓ **`Razionale r(n,d);`**

Costruttore generico a due argomenti che inizializza un `Razionale` avente n come numeratore e d come denominatore. Il denominatore non può essere zero, quindi se il programma tenta di creare un `Razionale` con denominatore zero verrà invocata la funzione `exit(1)`.

✓ **`cout << r;`**

Operatore di uscita per il tipo `Razionale`. Esso stampa il numero nel seguente formato.

`3/4`

`5` ← Per i razionali che sono interi

`-4/7` ← Per i razionali negativi

I numeri razionali vengono sempre stampati in forma semplificata, quindi non verrà mai prodotto in uscita il formato `5/10`, ma piuttosto `1/2`.

✓ **`r+s;`**

✓ **`r-s;`**

✓ **`-r;`**

✓ **`r*s;`**

✓ **`r/s;`**

Operatori aritmetici di base su oggetti `Razionale`. Funzionano come ci si aspetta che funzionino. Per l'operatore di divisione si intende la divisione esatta (senza resto), che dà come risultato a sua volta un `Razionale`. Se il programma tenta di dividere per zero verrà invocata la funzione `exit(1)`.

✓ **`r<s;`**

Operatore di minore stretto su oggetti di tipo `Razionale`. Funziona come ci si aspetta che funzioni.

--- Metodi invocati nella SECONDA PARTE di main.cpp: ---

✓ **`~Razionale();`**

Qualora sia necessario, implementare il distruttore.

✓ **`ordina(v, n);`**

Metodo globale che ordina un vettore di oggetti `Razionale v` lungo `n` in ordine crescente.

✓ **`div_int(a, b, q, r);`**

Metodo globale che calcola la divisione euclidea (cioè la divisione intera) tra il `Razionale a` ed il `Razionale b`, restituendo il quoziente in `q` ed il resto in `r` tali che `q*b+r` dia `a`. Nonostante il quoziente sia per definizione intero, `q` è comunque restituito come un `Razionale`. Anche qui, se il programma tenta di dividere per zero verrà invocata la funzione `exit(1)`. Nel caso il dividendo e/o il divisore siano negativi, la divisione euclidea segue le regole del C, ovvero: (i) il valore assoluto del resto deve essere sempre minore del valore assoluto del divisore; (ii) il segno del quoziente deve essere positivo se e solo se divisore e dividendo sono concordi; (iii) il segno del resto deve essere concorde con il segno del dividendo.

Ecco alcuni esempi di divisione euclidea tra razionali:

`5/8` diviso `2/13` uguale `4` resto `1/104`;

`-5/8` diviso `2/13` uguale `-4` resto `-1/104`;

`13/20` diviso `-2/7` uguale `-2` resto `11/140`;

`-13/20` diviso `-2/7` uguale `2` resto `-11/140`.

Nota: un'implementazione ottimizzata senza cicli né ricorsioni verrà valutata maggiormente.

✓ **`max_occhio(v, n);`**

Metodo globale che calcola il massimo di un vettore di oggetti `Razionale v` lungo `n` secondo un metodo "a occhio". Tale metodo segue le seguenti regole: (i) per i razionali negativi o zero nessuno è maggiore dell'altro (sono tutti uguali) e tutti sono minori di quelli positivi; (ii) per i razionali positivi, è maggiore quello che nella forma semplificata ha il denominatore minore; (ii) a parità di denominatore, è maggiore quello che nella forma semplificata ha il numeratore maggiore. Nel caso in cui tutti gli elementi siano negativi (e quindi tutti uguali secondo il confronto a occhio) verrà restituito il primo. Se il programma tenta di calcolare il massimo a occhio di un vettore vuoto verrà invocata la funzione `exit(1)`.

Mediante il linguaggio C++, realizzare il tipo di dato astratto definito dalle precedenti specifiche. Non è permesso l'utilizzo di variabili globali, né delle funzionalità della libreria STL come il tipo `string`, il tipo `vector`, il tipo `list`, ecc. **Gestire le eventuali situazioni di errore**.

USCITA CHE DEVE PRODURRE IL PROGRAMMA

```
--- PRIMA PARTE ---
Test dei costruttori ed operatore di uscita:
0 1/2 1/2 5/8
Test delle operazioni aritmetiche
1/2 1/2 1 9/8
-1/2 1/2 0 0 1/8
0 -1/2
0 0 1/4 9/16
0 1 7/4
Test con razionali negativi
-2/13 -1/3 -1
1/13 -81/104 -53/78
Test con operatore <
100111

--- SECONDA PARTE ---
Test dell'(eventuale) distruttore
(distruttore invocato)
Test con max_occhio
3
Test di div_int
1 0
4 1/104
-4 1/104
3 -1/26
-1 -7/24
Test di ordina
-5/2 -5/2 0 10/23 3/4 7/8 1 12/5 21/8 3
```

Note per la consegna:

Affinché l'elaborato venga considerato valido, il programma **deve** produrre almeno la prima parte dell'output atteso. In questo caso, i docenti procederanno alla valutazione dell'elaborato **solo se** lo studente avrà completato l'autocorrezione del proprio elaborato.
In **tutti** gli altri casi (per esempio, il programma non compila, non collega, non esegue o la prima parte dell'output non coincide con quella attesa), l'elaborato è considerato **insufficiente** e, pertanto, **non verrà corretto**.
